RECEIVED
CENTRAL FAX CENTER
JAN 18 2008

AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently amended) A method of enabling execution of data-relative code within a non data-relative environment, including the steps of:
   i) locating one or more instances of the use of a data-relative offset within a module of a code segment within the non data-relative environment;
   ii) calculating a new offset independent of a data segment; and
   iii) replacing the data-relative offset with the new offset in the code segment module;
   steps (i) to (iii) occurring outside the runtime of an application using the code segment.

2. (Original) A method as claimed in claim 1 wherein the data-relative offset is used for computing the target address of a branch instruction.

3. (Currently Amended) A method as claimed in claim [[2]]1 wherein the new offset is relative to [[the]]an instruction pointer.

4. (Currently Amended) A method as claimed in claim 3 wherein the new offset is calculated ~~using the formula~~in accordance with:
   new_offset = link_time_data_address + data_relative_offset –
   (link_time_module_start + instruction_pointer_offset);
   wherein:
   new_offset = the new offset;

link_time_data_address = the data address during link time;
data_relative_offset = the address relative to the data address;
link_time_module_start = the address of the start of the module of the code segment during link time; and
instruction_pointer_offset = the offset, relative to the start of the code segment module, of the instruction that calculates the new address using the instruction pointer (IP).

5. (Original) A method as claimed in claim 3 wherein steps (i) to (iii) occur during runtime of an application using the code segment.

6. (Original) A method as claimed in claim 5, including the step of:
iv) executing the modified code segment module.

7. (Original) A method as claimed in claim 6 wherein the code segment module is executed on an HP-UX platform.

8. (Original) A method as claimed in claim 6 wherein the code segment is compiled for a little-endian system and the code segment module is executed on a big-endian system.

9. (Original) A method as claimed in claim 6 wherein the code segment module is executed on a non-native platform.

10. (Original) A method as claimed in claim 5 wherein the new offset is an absolute runtime address.

11. (Currently Amended) A method as claimed in claim 10 wherein the absolute runtime address is calculated ~~using the formula~~in accordance with:

runtime_addr = link_time_data_addr + data_relative_offset – link_time_code_segment_start + runtime_code_segment_start;
wherein:
runtime_addr = the absolute runtime address;
link_time_data_addr = the data address during link time;
data_relative_offset = the address relative to the data address;
link_time_code_segment_start = the address of the start of the code segment during link time; and
run_time_code_segment_start = the absolute address of the start of the code segment during runtime.

12. (Cancelled)

13. (Currently Amended) A method as claimed in claim [[12]]1, including the step of:

v) saving the modified code segment module to non-volatile memory.

14. (Original) A method as claimed in claim 1 wherein the code segment module is within a shared library.

15. (Currently Amended) A method as claimed in claim 14, including the step of:

vi) copying the code segment module to modifiable memory;

wherein step (vi) occurs before step (iii) and wherein during step (iii) the data-relative offset is replaced in the copied code segment module.

16. (Original) A method as claimed in claim 15, including the step of:

vii) allocating the modifiable memory read, write and execute permissions;

wherein step (vii) occurs before step (vi).

17. (Original) A method as claimed in claim 14 wherein the code segment within the shared library is mapped as writable.

18. (Original) A method as claimed in claim 14 wherein the code segment module is one selected from the set of an .init section from the shared library and a .fini section from the shared library.

19. (Original) A method as claimed in claim 1 wherein the code segment is within an application.

20. (Currently Amended) A method as claimed in claim 1 wherein the code segment was compiled ~~using~~ by a compiler which inserts code that uses data–relative offsets.

21. (Cancelled)

22. (Original) A method as claimed in claim 1 wherein a dynamic loader performs all the steps.

23. (Original) A method as claimed in claim 2 wherein the data-relative offset is located in step (i) by backtracing the target register of the branch instruction.

24. (Original) A method as claimed in claim 1 wherein the code segment is a Linux code segment.

25. (Currently Amended) A system for enabling execution of data-relative code within a non data-relative environment, including:

[[(i)]] A processor arrangement adapted to (a) locate the use of data-relative offsets within a module of a code segment, (b)[[to]]

calculate a new offset independent of a data segment, and (c)[[to]] replace the data-relative offset with the new offset in the code segment module; the processor being arranged to perform (a), (b) and (c) outside the runtime of an application using the code segment; and

[[(ii)]] Memory adapted to store the code segment module.

26. (Currently Amended) A system as claimed in claim 25 wherein the ~~data-relative offset is used~~processor is arranged for computing the target address of a branch instruction in response to the data-selective offset.

27. (Currently Amended) A system as claimed in claim 26 wherein the processor is arranged for ascertaining the new offset [[is]]relative to the instruction pointer.

28. (Currently Amended) A system as claimed in claim 27 wherein the processor is arranged to calculate the new offset ~~is calculated usingin~~ accordance with the formula:

new_offset = link_time_data_address + data_relative_offset – (link_time_module_start + instruction_pointer_offset);

wherein:

new_offset = the new offset;

link_time_data_address = the data address during link time;

data_relative_offset = the address relative to the data address;

link_time_module_start = the address of the start of the module of the code segment during link time; and

instruction_pointer_offset = the offset, relative to the start of the code segment module, of the instruction that calculates the new address using the instruction pointer (IP).

29. (Original) A system as claimed in claim 25 wherein the processor is further adapted to execute an application using the modified code segment module.

30. (Original) A system as claimed in claim 29 wherein the new offset is an absolute runtime address.

31. (Currently Amended) A system as claimed in claim 30 wherein the processor is arranged to calculate the absolute runtime address ~~is calculated using~~in accordance with the formula:
runtime_addr = link_time_data_addr + data_relative_offset – link_time_code_segment_start + runtime_code_segment_start;
wherein:
runtime_addr = the absolute runtime address;
link_time_data_addr = the data address during link time;
data_relative_offset = the address relative to the data address;
link_time_code_segment_start = the address of the start of the code segment during link time; and
run_time_code_segment_start = the absolute address of the start of the code segment during runtime.

32. (Original) A system as claimed in claim 25, including:
iii) non-volatile memory adapted to store the modified code segment module.

33. (Currently Amended) A system as claimed in claim 25 wherein the processor is arranged to obtain the code segment [[is]]from within a shared library.

34. (Currently Amended) A system as claimed in claim 33, including:
iv) modifiable memory adapted to stored the code segment module;

and wherein the processor is further adapted to (a) copy the code segment module to the modifiable memory and ~~wherein~~ (b) replace the data-relative offset ~~is replaced~~ in the copied code segment module.

35. (Original) A system as claimed in claim 34 wherein the processor is further adapted to allocate the modifiable memory read, write and execute permissions.

36. (Original) A system as claimed in claim 33 wherein the code segment module is one selected from the set of an .init section from the shared library and a .fini section from the shared library.

37. (Original) A system as claimed in claim 25 wherein the code segment is within an application.

38. (Currently Amended) A system as claimed in claim 25 ~~wherein~~ further including a compiler for compiling the code segment ~~was compiled using a compiler which inserts~~and for inserting into the code segment code that uses data-relative offsets.

39. (Cancelled)

40. (Original) A system as claimed in claim 25 wherein the code segment is a Linux code segment.

41. (Currently Amended) A system as claimed in claim 29 ~~wherein the code segment module is executed on~~further including an HP-UX platform for executing the code segment module.

42. (Currently Amended) A system as claimed in claim 29 wherein the platform is arranged for compiling the code segment ~~is compiled~~ for a little-endian system and further including ~~the code segment module is executed on~~ a big-endian system for executing the code segment module.

43. (Currently Amended) A system as claimed in claim 29 ~~wherein~~ further including ~~the code segment module is executed on~~ a non-native platform for executing the code segment module.

44. (Original) A code segment module modified by the method of claim 1.

45. (Currently Amended) A binary file ~~containing~~ including a code segment module modified by the method of claim 1.

46. (Cancelled)

47. (Currently Amended) Storage media including computer readable[[the]] software ~~as claimed in claim 46~~for performing the method of claim 1.

48. (Original) A computer system for effecting the method of claim 1.

49. (Currently Amended) A memory storing computer readable[[the]] software ~~of claim 46~~for performing the method of claim 1.

50. (New) The system of claim 25 in combination with the non-data relevant environment.